REMARKS:

Reconsideration and allowance of the claims in the application are requested.

Applicant's attorney thanks the Examiner for the courtesy of an Interview conducted November 20, 2001. At the Interview, Applicant's attorney noted that neither Woster 5,982,946 nor Moore 6,189,046 B1 address the problem of software fault tolerance for software in which an error occurs. Woster describes accessing a server object database for a server object capable of performing a requested service for a client/server. Moore describes communication protocols, which allow client/server to communicate effectively. Moore does not aid a worker skilled in the art to expand Woster to perform the functions of the present invention. The Interview is summarized in the Interview Summary dated November 20, 2001.

Claims 1-16 are in the case. Claims 1-16 have been rejected under 35 USC 103(a) as unpatentable over USP 5,892,946 to G.W. Woster et al. issued April 6, 1999, filed September 12, 1995 (Woster) in view of USP 6,189,046 to K.E. Moore et al. issued February 13, 2001 and filed March 27, 1997 (Moore).

Before responding to the rejections, Applicant would like to distinguish Woster and Moore from the present invention (WANG), as follows:

1. Woster discloses a distributed computer system for telephone applications, which require redundant hardware and software services in a multi-site distributed object management environment. The environment consist of nodes and geographically remote locations in a global telephone telecommunication network, which must be able to quickly adjust to server outages at any point in the network. The distributed object messaging system, which. provides for the plurality of nodes, each node executing a plurality of processes. The processes register a plurality of objects in each node. The objects include client objects and server objects. The

server object database is used in each node to store a server object description for each server object registered in the node and objects registered in remote nodes. A client server interface is accessible by client objects and receives requests for services. The client server interface access the server object database for at least one destination server object capable of performing the client's objects requested service and forwards the service request to the destination service object at a local site or a remote site. Server object database is accessed to determine if the server object resides in the node name specified. Not found, a Distributed Object Environment (DOM) system declares an object not found error. If the server instance name is specified, the server object database is searched to ensure that the instance resides in the specified node name. An instance not found error is returned if the exact instance does not exist in the node. If the client specifies an object name, and does not request a specific node or instance, dome construction instance list of all instances and all nodes of the requested object name. The service request is sent to one or more nodes, which qualify and if no node qualifies, an error return is made to the client. If the client does not specify a specific node or instance, dome constructs a list of all nodes, which contain the object name. If none are found, an error is returned to the client. See col. 6, line 60, continuing to col. 7, line 45. Woster fails to disclose elements of Wang, as follows:

A. Woster discloses a distributed computing architecture linked by a distributed object management environment, which allows replacement of a client server sub-routine from another node and the computing system. *In contrast, Wang discloses a distributed component object model software architecture for software fault tolerance during and after failure of software in which an error occurs in the software.*

B. Woster discloses determining the presence or absence of a client or server object in response to a client request and sending an error message if an instant does not exist at a node. *In contrast, Wang detects an error in the object instance not the presence or absence of the object or instance.*

C. Woster constructs a list of all nodes, which contain an object instance; deletes from the list any node which does not have the highest service date and selects a node from the remaining list of nodes based on round-robin selection. *In contrast, Wang detects whether an error occurred in an instance; determines whether the error is minor and continues the instance, if minor, or if significant, invokes a second object instance from a pool to replace the first object instance.*

2. Moore discloses a communications infrastructure, which proivdes a mechanism which supports multiple simultaneous communication protocols enabling an application program executing on one process to make method calls on objects located in other processes and yet be totally oblivious to the communication's protocol used to deliver data between the two processes. A communication framework presents application code with an abstraction layer, included a distributed apply function. The apply function allows application programs to be written with any direct reference to the communication protocol selected implement a distributed supply function. The extraction layer further includes mechanism for causing self-marshalling and de-marshalling of arguments provided to remote procedures. The marshalling of arguments is accomplished in a manner that does not require knowledge of a memory layout chosen by the application and permits in-memory representations of abstract data-types to be independent of the underlying communications protocol. Moore fails to disclose elements of Wang, as follows:

A. Moore discloses communication protocols, which allow programs executing on one process to make method calls on objects in other processes, regardless of the communication protocols to deliver data between the processes. *In contrast, Wang discloses software fault tolerance methods for detecting and replacing software in a pool with other software in the pool.*

B. Moore attempts to invoke a method of a target object using a confirmed protocol entry, and, if not successful, using an unconfirmed entry. *In contrast, Wang discloses a registry after the control system control manager has failed to locate any object instance. The registry maps the program I.D. to the pathname of the server that supports the program I.D. See page 18, lines 10-14.*

C. Moore discloses self-marshalling and de-marshalling of arguments, which does not require knowledge of the memory layout chosen by an application. *In contrast, Wang employs marshalling as a part of enabling automatic client reconnection upon a server failure. See page 21, lines 1-13.*

D. Moore discloses creating a table of binding information hints, where each entry in a table includes information to be used to establish a connection to an object referred to by a virtual process using a transport protocol. *In contrast, Wang discloses a source code transparent wrapper intercept, intercepting normal connection requests sent by a client and issuing reconnection when a failure occurs, the implementation of dynamic wrappers allowing server objects to decide when to apply which wrapper, based on run-time information.*

Summarizing, Woster and Moore alone or in combination fail to show or suggest a distributed component object computing system, a server object instance, a pool of server object instances, where a server object instance invoked by a client functions until an error occurs in an

instance, which is detected by the instance or the client or the server and replaced and continued if the error is minor or replaced if the error is significant by invoking a second server object instance from the pool. Woster only discloses a distributed computing system, which replaces an instance when fail over occurs by checking a database to determine the presence or absence of a replacement for the failed instance. There is no disclosure in Woster of a client in an instance where a client identifies the error for purposes of replacement. Moore provides protocols for communication among object instances regardless of system architecture and does not address the missing element in Woster. Accordingly, a worker skilled in the art does not have the teachings in Woster and Moore, alone or in combination to implement the present invention of detecting an error in an object instance and in determining whether to continue the object instance or replace the object instance by another object instance as described in the present invention. Without such teaching the rejection of claims 1-16 is without support and should be withdrawn. Allowance of Claims 1-16 is requested.

Claims 1, 4, 11, 12-16 have been amended to clarify the invention for a better understanding thereof relative to the prior art. Rejection of claims 1-16 has been traversed.

Now turning to the rejections, Applicant provides responses to the indicated paragraphs of the Office Action, as follows:

REGARDING PARAGRAPHS 1/2:

A. Claim 1:

Claim 1 includes elements not shown in Woster or Moore, as follows:

(i) "... a client maintaining a pool of server object instances on multiple machines;"

The Examiner admits that Woster fails to disclose a pool of server objects where a client contact locate replacement server objects. See page 21, lines 19-21. Woster discloses a server object database is used in each node to store a server object description for each server object registered in the node. See col. 1, lines 56-60. There is no disclosure that the database describes pool of server object instances, which may be substituted for one another. Nor does Moore disclose a pool of object instances, which may be substituted for one another.

(ii) " a client invoking a first server object from the pool with a client request until an error occurs in the first server object instance;"

Woster discloses a distributed object management system declaring an error when an object instance is not located in the server object database, and not an error occurring in the software. See col. 6, lines 60-65. Nor does Moore disclose invoking an object instance until an error occurs in the object instance.

(iii) "determining whether the error is minor and continuing the first server object instance for if the error is significant the client invoking a second server object instance from the pool to replace the first server object instance"

Neither Woster nor Moor detect an error in an object instance; deleting the error and invoking a different server object instance by continuing the object instance if the error is minor or invoking a second server object instance running along one of the multiple machines as described in Fig. 18 and in the specification at page 25, lines 1-16.

Applicant submits that neither Woster nor Moore, alone or in combination, disclose or suggest the elements discussed in items (i), (ii) or (iii) above. Without such disclosure, there is no basis for the rejection of claim 1 under 35 USC 103(a). Withdrawal of the rejection and allowance of claim 1 are requested.

B. Claim 2:

"...wherein the error by the first server object instance is identified by the first server object instance."

Applicant can find no disclosure in Woster or Moore relating to the object instance detecting an error as shown in Fig. 18 described above. Without such disclosure in Woster and Moore, alone or in combination, there is no basis for the rejection of claim 2 under 35 USC 103(a). A worker skilled in the art has no teaching to implement claim 2.

C. Claim 3:

Claim 3 is patentable on the same grounds as claim 2. Withdrawal of the rejection of claim 3 is requested.

D. Claims 4-6:

(i) "...a determining program module that determines where the error is minor and continue with the first server object instance or is the error is significant the client invokes a second server object instance from the pool to replace the first server object instance."

Applicant can find no disclosure in Woster or Moore, alone or in combination, of a program module detecting whether the error requires invoking a different server object instance as shown in step 1507 and Fig. 18 and described in the specification at page 25, line 9 and continuing to line 16.

Without a disclosure in Woster or Moore relating to the determining program module, there is no basis for a worker skilled in the art, to implement a system of software tolerance and a distributed component object model. Withdrawal of the rejection of claims 4-6 under 35 USC 103(a) and allowance of claims 4-6 are requested.

E. Claim 7:

(i) "...determining a number (N) of different server object instance required to achieve a desired reliability or availability of a client;"

Applicant can find no disclosure in Woster or Moore using designed diversity as an approach to software fault tolerance, as described in the specification at page 22, line 20 and continuing to page 23, line 15. Moore only discloses replacing failover software without executing different programs implementing the same functionality in selecting a majority result for the selected program.

(ii) "...maintaining a pool of at least N different server object instances on multiple machines; and invoking at least end different server object instances form the pool."

Applicant can find no disclosure in Woster or Moore relating to invoking different end-server object instances from the pool.

Without such disclosure, there is no basis for the rejection of claim 7 under 35 USC 103(a). A worker skilled in the art, without such teaching, would have no basis for implementing the method of claim 7.

F. Claim 8-11:

(i) "Identifying an error produced by one of the N different server object instances'"

Applicant can find no disclosure in Woster or Moore relating to detecting or identifying an error in one of N-different server object instances. Both Woster and Moore fail to disclose or suggest a diversity design approach for fault tolerance.

(ii) "...determining that an error was produced by one of the N-server object instances and removing the server object instance that produced the error, if N is 3 or greater."

Applicant can find no disclosure in Woster or Moore relating to detecting an error in one of the N-different server object instances and removing the server object instance that produced the error, if the number of different server object instances is 3 or greater.

Without a disclosure in Woster or Moore relating to design diversity for software fault tolerance, there is no basis for a worker skilled in the art to implement Claim 8 - 11. Withdrawal of the rejection of Claims 8 - 11 under 35 USC 103(a) and allowance thereof are requested.

G. Claims 12-13:

Claims 12-13 track claims 7 and 11 and are believed patentable on the same basis as claims 7-11. Withdrawal of the rejection of claims 12-13 under 35 USC 103(a) and allowance thereof are requested.

H. Claims 14-16:

Applicants submit that Woster and Moore, alone or in combination, fail to disclose: "a detecting program module which detects whether one of the end-server object instances produces an error;" or "an error identifying program module which enables the client to identify an error produced by one of the end-server object instances" and "an error identifying module with one of the end-different server object instances to identify an error produced by one of the end-different server object instances whereby the client invokes another server object instance from the pool as a replacement from the server object instance."

Applicant submits the absence of a disclosure or suggestion in Woster or Moore, alone or in combination, relating to the above elements does not provide a basis for a worker skilled in the art to implement the system of Claims 14-16. Withdrawal of the rejection of claims 14-16 under 35 USC 103(a) and allowance thereof are requested.

REGARDING PARAGRAPH 3:

Applicant has reviewed the prior art of record and concludes that Davidson 6,042,614; Whitehead, 6,085,030 and Arunachalam 6,212,556 fail to disclose the elements of claims 1-16 and are accumulative to the cited art.

CONCLUSION:

Having amended the claims to clarify the invention for a better understanding thereof; distinguished claims 1-16 from the cited references and considered the cited, but not applied art, Applicant requests entry of the amendment, allowance of the claims and passage to issue of the case. Pursuant to 37 C.F.R. § 1.121, Attachment A, showing a mark-up version of the changes made to the specification and claims by the current Amendment is attached hereto.

AUTHORIZATION:

The Commissioner is hereby authorized to charge any additional fees which may be required for the timely consideration of this amendment under 37 C.F.R. §§ 1.16 and 1.17, or credit any overpayment to Deposit Account No. 13-4503, Order No.. 2455-4502US1.

Respectfully submitted,
MORGAN & FINNEGAN, L.L.P.

Dated: November 26, 2001

By: Joseph C. Redmond

Joseph C. Redmond, Jr.
Registration No. 18,753
202-857-7887 – Telephone
202-857-7929 – Facsimile

CORRESPONDENCE ADDRESS:

Morgan & Finnegan L.L.P.
345 Park Avenue, New York
New York 10154

PATENT
Attorney Docket No. 2455-4502US1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	WANG	:	
Serial No.:	09/137,907	:	Group Art Unit: 2184
Filed:	August 21, 1998	:	Examiner: D. Le

For: METHOD AND SYSTEM FOR PROVIDING RELIABILITY AND AVAILABILITY IN A DISTRIBUTED COMPONENT OBJECT MODEL (DCOM) OBJECT ORIENTED SYSTEM

Commissioner for Patents
Washington, D.C. 20231

ATTACHMENT A – SHOWING A MARKUP OF THE CHANGES

Sir:

IN THE CLAIMS:

Claims 1, 4, 11, 12, 13, 14 15, and 16 have been AMENDED as follows:

1. (Amended) A method for enhanced software fault tolerance in a Distributed Component Object Model comprising the steps of:

a client maintaining a pool of server object instances on multiple machines;

the client invoking a first server object instance from the pool with a client request **until an error occurs in the first server object instance**; and

determining whether the error is minor and continuing the first server object instance or if the error is significant the client invoking a second server object instance from the pool based **[on an error by] to replace** the first server object instance in processing the client request.

4. (Amended) A system for enhanced software fault tolerance in a Distributed Component Object Model comprising:

a first server object instance from the pool with a client request **until an error occurs in the first server object instance**; and

a determining program module which determines whether the error is minor and continuing the first server object instance or if the error is significant [; **and**] the client invokes a second server object instance from the pool **[based on an error by]** **to replace** the first server object instance in processing the client request.

11. (Amended) The method of claim 7 further comprising the steps of:

comparing processing results of the N different server object instances to detect errors in any one of the N different server object instances;

determining that an error was produced by one of the N different server object instances; and

removing the server object instance that produced the error, if N is three or greater.

12. (Amended) A system for enhanced software fault tolerance in a Distributed Component Object Model comprising:

a client; **[and]**

a pool of at least N different server object instances on multiple machines, wherein N different server object instances are required, the client maintains the pool, and the client invokes the N different server object instances from the pool to provide software fault tolerance**; and**

a comparing program module which compares processing results of the N different server objects to detect errors in any one of the N different server object instances.

13. (Amended) The system of claim 12 wherein the system comprises:

a determining program module which determines that an error was produced by one of the N different server object instances **[and] whereby** the server object instance that produced the error is removed from the system, if N is three or greater.

14. (Amended) A system for enhanced software fault tolerance in a Distributed Component Object Model comprising:

a client; **[and]**

a pool of at least N plus one different server object instances on multiple machines, wherein N different server object instances are required, the client maintains the pool, the client invokes N different server object instances from the pool **[,]; and**

a detecting program module which detects whether one of the N server object instances produces an error **[and] whereby** the client invokes another server object instance from the pool as a replacement for the server object instance that produced the error.

15. (Amended) A system for enhanced software fault tolerance in a Distributed Component Object Model comprising:

a client; **[and]**

a pool of at least N plus one different server object instances on multiple machines, wherein N different server object instances are required, the client maintains the pool, the client invokes N different server object instances from the pool**[,]** **;and**

an error identifying program module which enables the client **[identifies]** **to identify** an error produced by one of the N different server object instances **[and]** **whereby** the client invokes another server object instance from the pool as a replacement for the server object instance that produced the error.

16. (Amended) A system for enhanced software fault tolerance in a Distributed Component Object Model comprising:

a client; **[and]**

a pool of at least N plus one different server object instances on multiple machines, wherein N different server object instances are required, the client maintains the pool **[, the client]** **and** invokes N different server object instances from the pool**[,]** **; and**

an error identifying program module which enables one of the N different server object instances **[identifies]** **to identify** an error produced by one of the N different server object instances **[and]** **whereby** the client invokes another server object instance from the pool as a replacement for the server object instance that produced the error.